QualiPSo

Quality Platform for Open Source Software

IST- FP6-IP-034763

Working document wd 5.6.1

Experimentation on the trustworthiness of Open Source Software, version 3.0

Luigi Lavazza
Sandro Morasca
Davide Taibi
Davide Tosi

Due date of deliverable: 31/08/2010

Actual submission date: 31/01/2011

This work is partially funded by EU under the grant of IST-FP6-034763.

Change History

Version	Date	Status	Author (Partner)	Description
1.1	April 30, 2009	draft	Luigi Lavazza	Released for internal review by researchers from Univ. Insubria
1.2	May 30, 2009	draft	Luigi Lavazza	Released for review
2.0	June 30, 2009	final	Luigi Lavazza	Takes into account reviewers' comments. Released
3.0	January 31, 2011	final	Luigi Lavazza	Reports about the complete set of experiments, including those performed in the second iteration of work within WP5.6.

What is new

This document reports the experimentation activities carried out in the second iteration of the work, during the last reporting period of the project.

The second iteration of the work used tools that were made available at the end of the third year or during the fourth year of the project. Therefore, a good deal of the results reported here are new.

EXECUTIVE SUMMARY

Definition of Task 5.6.1 - Experimentation

In the Description of Work the task is defined as follows:

Empirical studies will be carried out in industrial environments. Therefore, experiments will need to be carefully planned, designed, and executed, to minimize the risk of having incomplete or misleading information. Clearly, the second iteration of this task in the second round of experiments will benefit from the experience gathered in the first round of experiments. The empirical studies will be as little invasive as possible for the industrial environments studied to disturb the observed environment as little as possible and also maximize the chances that data are actually collected from the industrial environment. To this end, the automated tools built in WP5.5 will be used. At any rate, questionnaires and interview may also be used to collect additional pieces of information that would not be possible to retrieve from the raw data. The collected information will be organized and stored in repositories. In the second iteration of this Task, some measures used in the first round of experiments may be deleted, while others may be added, based on the results of the first round of experiments. This task will clearly provide inputs to the tool building WP 5.5 and will rely on the tools to be carried out effectively and efficiently. In addition to data on trustworthiness, data on the cost-effectiveness of and practicality of the approach will be collected, to assess the overall impact that the approach may have on industrial environments.

Objectives

The goal of the task is to assess the effectiveness of the approach outlined in Activity A5. In particular, the trustworthiness factors identified in WP5.3, the test approaches, suites and benchmarks identified in WP5.4, and the tools developed, customized and integrated in WP5.5 are experimented with in Task 5.6.1.

The main result of the experimentation generates the data concerning the trustworthiness of the OSS products examined during the experimentation. These data are an input to Task 5.6.2, which analyzes them to find out whether the factors identified were actually influential on the trustworthiness of the OSS products and artefacts, and –if so– derives quantitative model that represent such dependency.

Other results of the task are feedbacks concerning the methods, models, techniques and tools being defined.

Method

The main instrument for the experimentation is represented by empirical studies and measurement.

According to the indications from WP5.3, the experimentation addresses two aspects of trustworthiness: the perception of trustworthiness by users and the contribution to trustworthiness from the qualities of the software products. The former is assessed by collecting evaluations from users (both from industry and public administrations); the second is measured.

Users evaluations are collected by means of questionnaires and interviews.

The measurements of the OSS product are performed using the tools identified, produced, or customized in WP5.5. The collected information is stored in repositories.

The main results obtained are:

- The definition of a GQM plan that is fully operational and can be used to support the trustworthiness measurement and analysis process.
- The data reporting the users' subjective perceptions of the trustworthiness of OSS product.
- A great deal of measures –all properly stored in a measure repository– concerning various features of OSS products:
    - Static code measures
    - Dynamic code measures.
    - Measures about the product versioning and configuration.
    - Measures about the licensing information provided with OSS products.

Document Information

IST Project Number	FP6 – 034763	**Acronym**	QualiPSo
Full title	Quality Platform for Open Source Software		
Project URL	http://www.qualipso.org		
Document URL			
EU Project officer	Michel Lacroix		

Deliverable	**Number**	wd 5.6.1 QualiPSo	**Title**	Experimentation on the trustworthiness of Open Source Software, version 3.0
Work package	**Number**	5.6	**Title**	Experimentation and model building
Activity	**Number**	A5	**Title**	Trustworthy Results

Date of delivery	**Contractual**	31/08/2010	**Actual**	31/01/2011
Status			final ☐	
Nature	Report ☑ Demonstrator ☐ Other ☐			
Dissemination Level	Public ☑ Consortium ☑			
Abstract (for dissemination)	This document reports about the practical experimentation of the techniques, methods and models defined in the QualiPSo project for the evaluation of the trustworthiness of open source software.			
Keywords	Trustworthiness, empirical studies, measurement.			
Authors (Partner)	Luigi Lavazza (University of Insubria)			
Responsible Author	Luigi Lavazza		**Email**	luigi.lavazza@uninsubria.it
	Partner	INS	**Phone**	+39-0332-219830

TABLE OF CONTENTS

1 THE BIG PICTURE

In order to make the rest of the document clearer, the work to be carried out in WP 5.6 is summarized here.

Figure 1 reports the conceptual model of the entities involved in the work. We start with a GQM measurement plan –defined in WP5.3– whose execution leads to the construction of the QualiPSo model of trustworthiness. The execution of the GQM plan involves two phases: the actual measurement (described in this document) and the analysis of the collected data (described in the various versions of WD 5.6.2).

In particular, the GQM plan involves two types of metrics: objective metrics, which are meant to measure the intrinsic, objective properties of the OSS products, and subjective metrics (named “subjective trustworthiness evaluations” in Figure 1), which are meant to represent how users (subjectively) perceive the trustworthiness of OSS products.

The actual measures corresponding to the GQM metrics definitions are collected and stored in a repository.

There is a set of measures for every considered OSS product.

The analysis phase that is described in WD 5.6.2 aims at correlating the objective, measurable properties of OSS products (like modularity, defect density, size, etc.) with their properties (like reliability, security, etc.) that are perceived by the users. Trustworthiness is the ‘sum’ of the subjective properties.

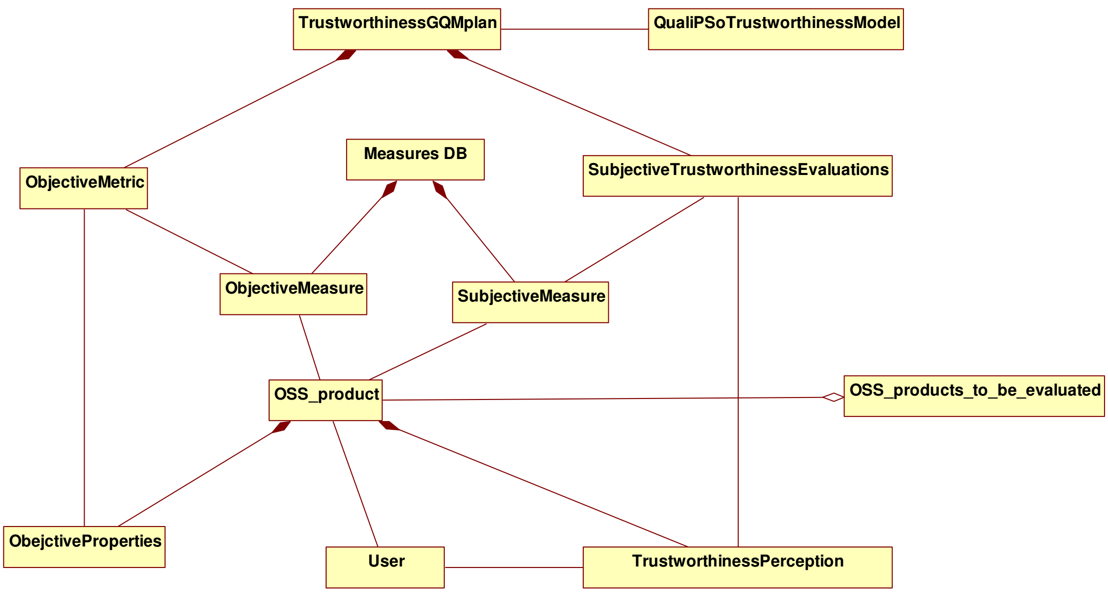

Figure 1. Conceptual model of the items involved in WP5.6.

A high level view of the process carried out in WP5.6 is reported in Figure 2. As already mentioned, the work starts with the definition (carried out in WP 5.3) of the GQM plan. The GQM plan, and the list of projects to be examined drives the collection of –subjective and objective– data. The collection of data is largely supported by tools (namely, those developed in WP5.5) but not completely automated, since a good deal of the required information can be safely retrieved only manually.

The collected data are analyzed and a tentative quantitative model of trustworthiness is derived. The data analysis activity will also possibly result in suggestions about the refinement, extension or reduction of the GQM plan. In fact, the work described in Figure 2 will be carried out in two subsequent phases.

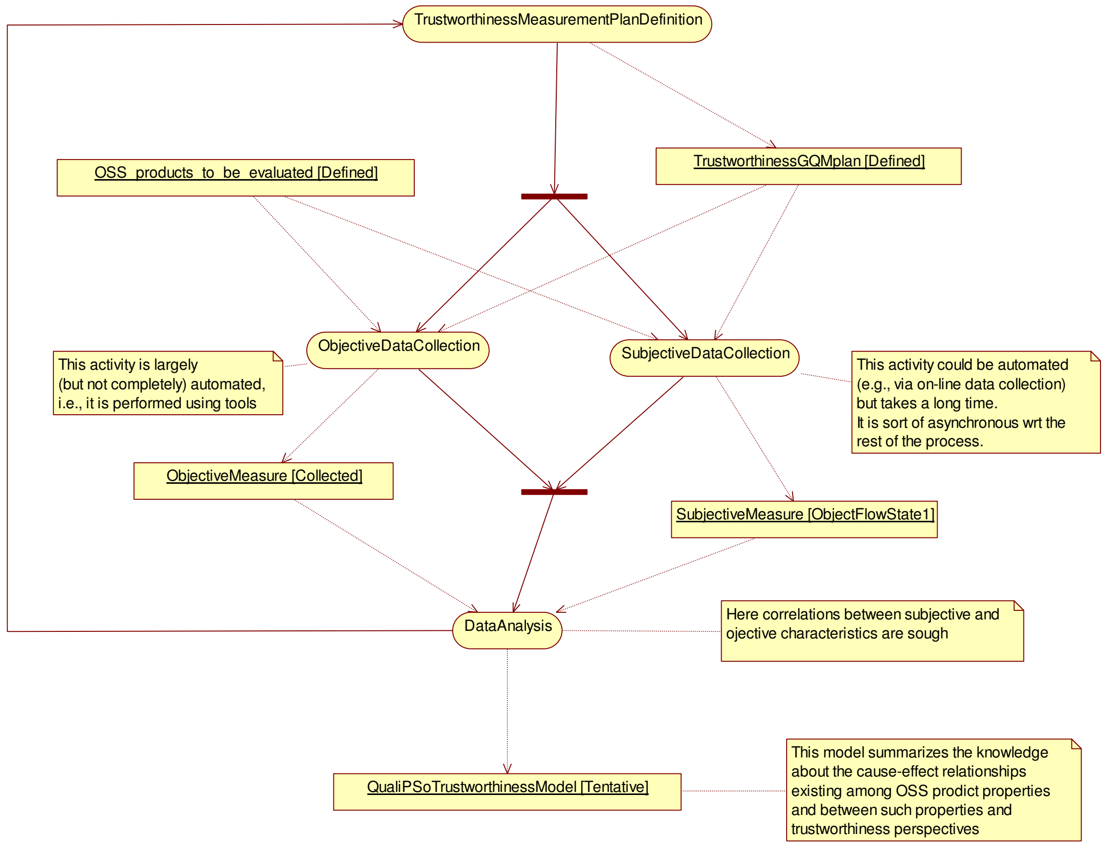

Figure 2. Workflow of activities in WP5.6.

2 THE OSS PRODUCTS BEING ANALYSED

In order to supply Task 5.6.2 with enough data point to derive statistically significant models, over 44 OSS products were chosen for evaluation, of which 22 written in Java and 22 written in C++ (the criteria used for the choice are reported in previous deliverables and working documents [7][17]).

The set of OSS products evaluated during the second round of experiments is reported in Table 1.

Table 1. The list of OSS products being evaluated during the first round of experiments

Java Product	C++ products
Checkstyle	Ant
Eclipse	Axis
Findbugs	BusyBox
Hibernate	CVS
HttpUnit	CygWin
Jakarta CommonsIO	DDD
JasperReport	GDB
JBoss	Gnu C Library
JFreeChart	Gnu GCC
JMeter	Lib XML
Log4J	Linux Kernel
PMDV	Mono
Saxon	MySQL
Spring-FW	OpeLDAP
ServiceMix	Open Pegasus
Struts	Open SSL
Tapestry	Perl
TPTPV	PosgreSQL
Velocity	SpiderMonkey
Weka	SQLite
Xalan	Subversion
Xerces	TCL/Tk

Figure 3. Role of the measures DB in WP5.6.

3 DEFINITION OF THE GQM PLAN

The “phase zero” of the experimentation consists in defining the GQM plan which provides guidance to the experimentation phase.

The definition of the GQM plan was supported by the usage of the GQM tool.

The definition of the plan starts with the definition of the GQM goal, according to the usual GQM paradigm (object, purpose, quality, viewpoint, environment): see Figure 4.

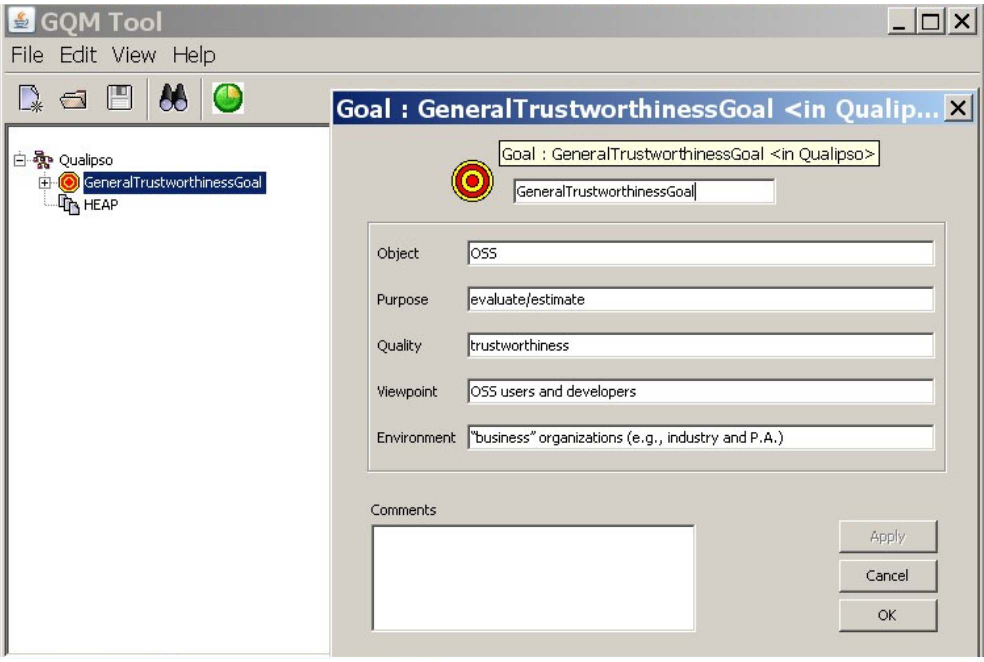

Figure 4. Definition of a GQM plan

The next step consists in defining the quality focuses and variation factors. This is done according to the conceptual definition of trustworthiness and the properties of software that are expected to affect it [11] (Figure 5).

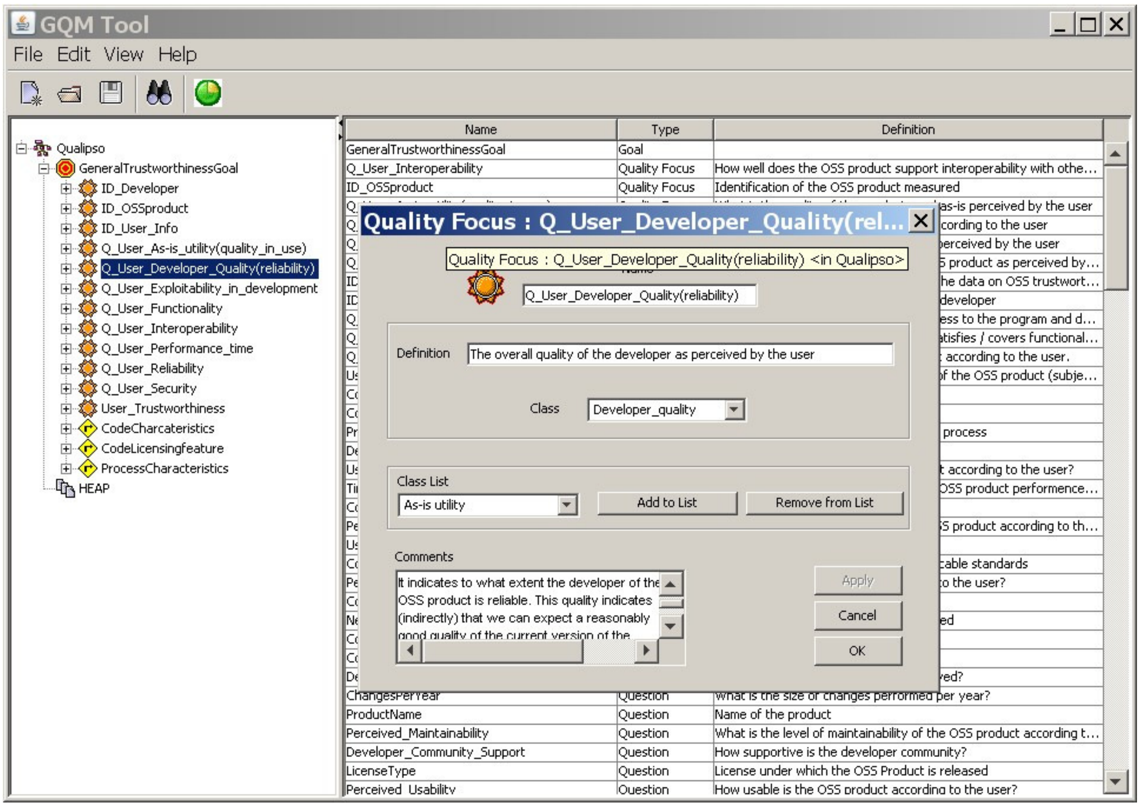

Figure 5. Definition of a quality focus

After quality foci and variation factors have been defined, they are refined into questions and metrics[1]. Figure 6 shows the definition of the metric “NumClasses”, which is one of the metrics refining question “CodeSize”, which belongs to variation factor “CodeCharateristics”.

It can be noticed that the tool allows the specification of the type of metric scale (absolute in Figure 6), the origin of the data (MacXim tool in Figure 6), and comments (yet to be written in Figure 6) to ease the comprehension and the maintenance of the plan.

It is important that the elements of the GQM plan be well specified, since they must match the needs of the investigation, be supported by tools, and be clearly understood by the analyzers.

[1] Actually, the definition proceeds in an iterative way, characterized by additions and deletions of GQM plan elements, according to the growing understanding of the problem at hand. Here we are showing the process as a sequence of steps to ease the presentation.

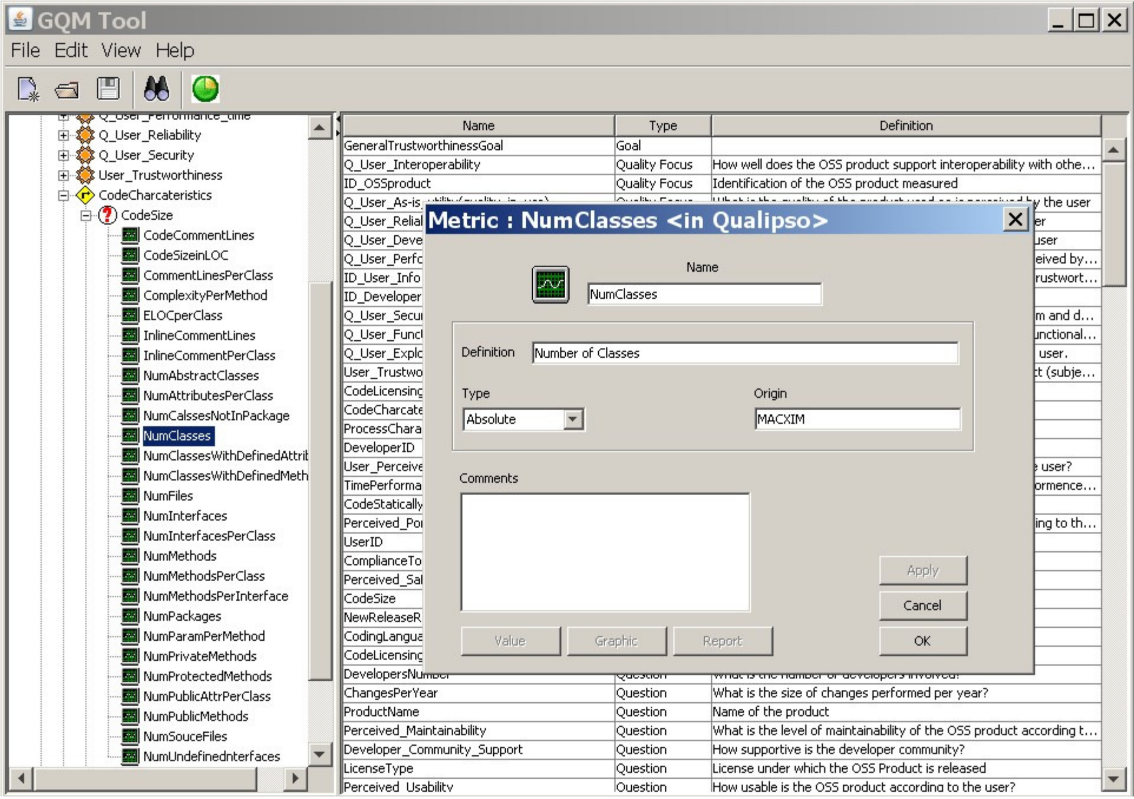

Figure 6. Definition of a metric

The GQM tool saves plans in a sort of XML format. In order to make the plans readable even without the GQM tool, a CSS file has been defined to support the visualization of GQM plans.

The GQM plan can then be visualized by means of any browser, as shown in Figure 7. Actually, the documentation reported in [11] was produced as described above.

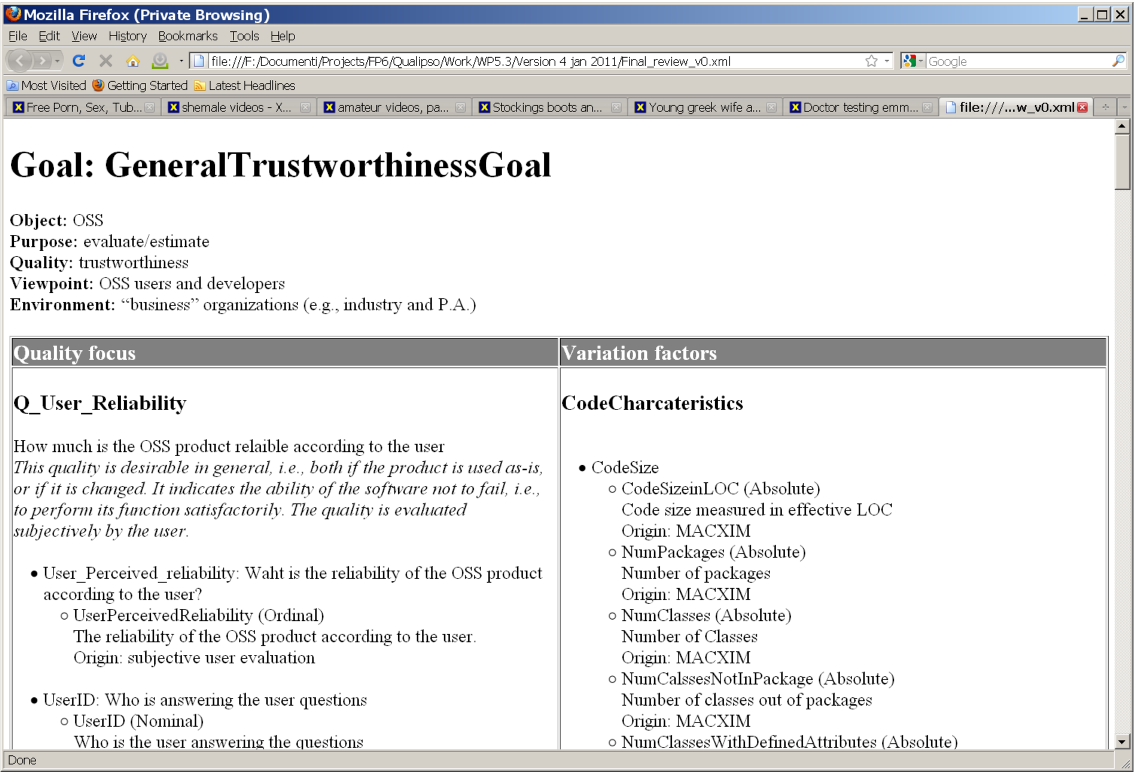

Figure 7. Visualization of the plan

4 SUBJECTIVE EVALUATION OF PERCEIVED TRUSTWORTHINESS

4.1 Approach

The collection of the subjective evaluations of the various aspects of trustworthiness by users was carried out with the help of a questionnaire. Users compiled the questionnaire in presence of QualiPSo people, so that any possible question or doubt about the questionnaire could be clarified.

The questionnaire had to concern:

- Multiple subjective qualities (as described in the GQM plan: see [8][9][10][11]).
- Multiple products, in order to support a statistically significant analysis.

Since every quality should ideally have been evaluated for every product, it was necessary to limit both the number of properties and products, to keep the time needed to fill the questionnaire reasonable.

To this end, the original version of the GQM plan was simplified a little: only top-level qualities were evaluated, and a few ones were just excluded from the questionnaire.

The final version of the questionnaire contained twelve questions about the products, and a few about the respondents. The questions concerned 22 Java programs and 22 C++ programs. The questionnaire is reported in the appendix (Section 10).

4.2 Results

Up to the beginning of October 2010, 694 questionnaires were collected. Overall, they account for 4101 evaluations (of which 1357 for Java projects and 2744 for C++ projects).

The questionnaires were collected at major international events, not necessarily dealing with OSS topics, as summarized in Table 2.

Table 2. Events where data were collected.

Event	Date (in year 2009) and location	Collected questionnaires	Product evaluations
Apache Conference	March 24-27 2009, Amsterdam, The Netherlands	15	31
OW2 Conference	April 1-2, 2009, Paris, France	20	31
XP 2009	April 24-30, 2009, Pula, Italy	12	95
OSS 2009	June 2-5, 2009, Skovde, Sweden	2	5
ICSE 2009	May 15-20, 2009, Vancouver, Canada	9	69
CONFSL 2009	June 12-13, 2009, Bologna, Italy	3	27
QualiPSo Meeting	July 1-22, 009, Madrid, Spain	6	38
ESC	August 30-31, 2009, Venice, Italy	31	411
FOSDEM	February 6-7, 2010, Brussels	145	967
XML Conf	March 13-15, 2010, Prague	42	168
Microsoft Real Code Conference	May 25, 2010, Firenze	18	86
CONFSL 2010	June 18-19, 2010, Cagliari	8	37
OSCON	July 2010, Portland (OR)	201	1034
Debian Conference	September 18-19, 2010, Perugia	19	107
Open World Forum	September 30 - October 1, 2010, Paris	149	894
OpenOpportunity	October 7-8, 2010, Castiglione del lago	5	49
FossA	November 8-10, 2010, Grenoble	7	37
Others		2	15

The number of evaluations collected per product is reported in Figure 8. The respondents were invited to declare their familiarity with the evaluated products. Figure 8 indicates also how many respondents were familiar with the OSS products. This is a relevant information: since evaluations by people with little familiarity with OSS products were excluded by the analysis carried out in Task 5.6.2.

Figure 8 also gives an idea of the relative popularity of the analyzed products.

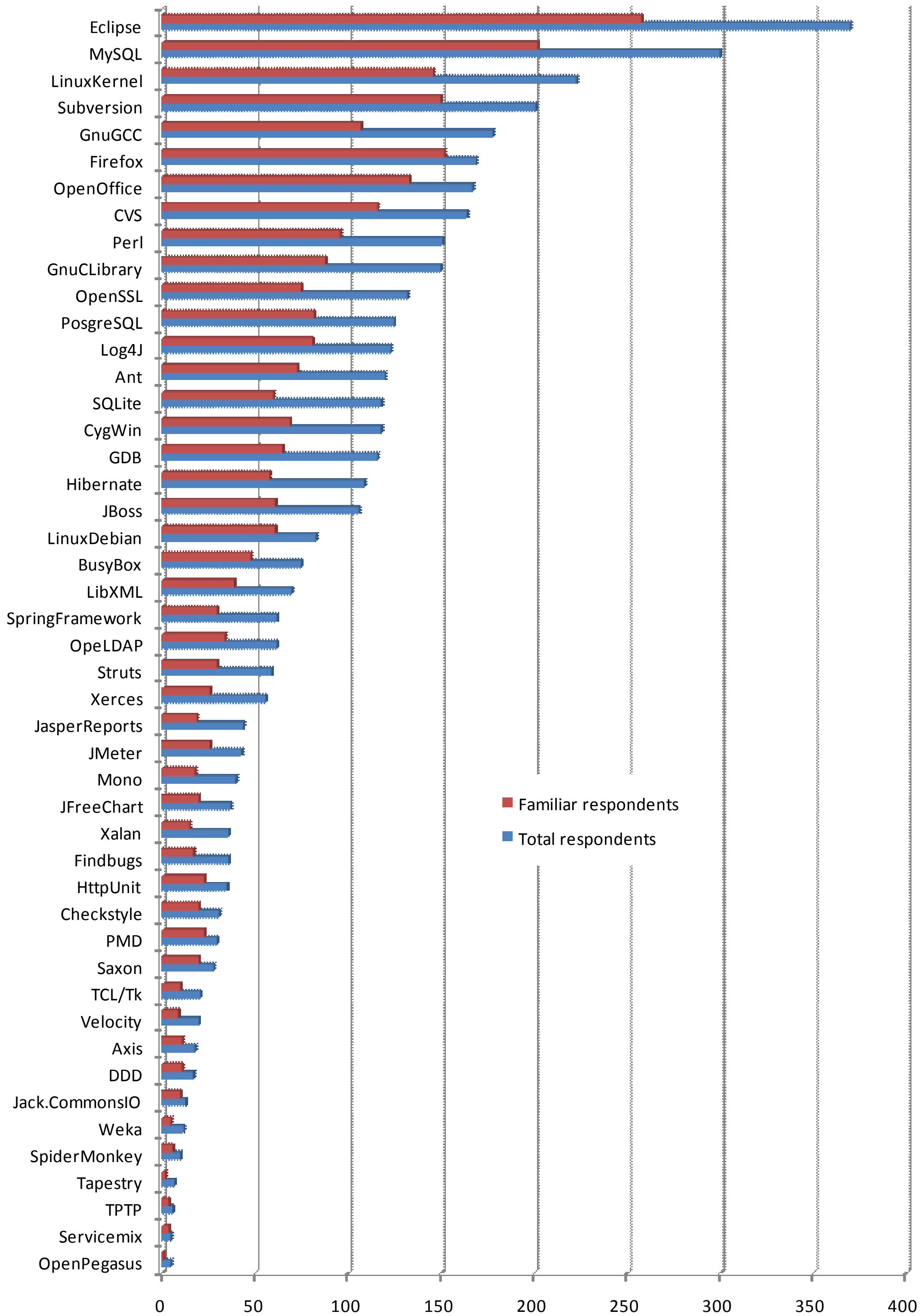

Figure 8. Total number of respondents and respondents with good familiarity, per OSS product

Even though subjective evaluations were mainly intended for analysis in Task 5.6.2, where the quantitative models of trustworthiness are derived, it interesting to look the subjective user evaluation alone, in order to understand how satisfied are users with OSS products.

In Figure 9 we reported the median of the fractions of satisfied users for each evaluated quality. Note that in this computation we considered satisfied the users that assigned grades 5 or 6, i.e., chose a relatively high threshold: if we had included also the moderately satisfied users, we would have reached higher median values, of course.

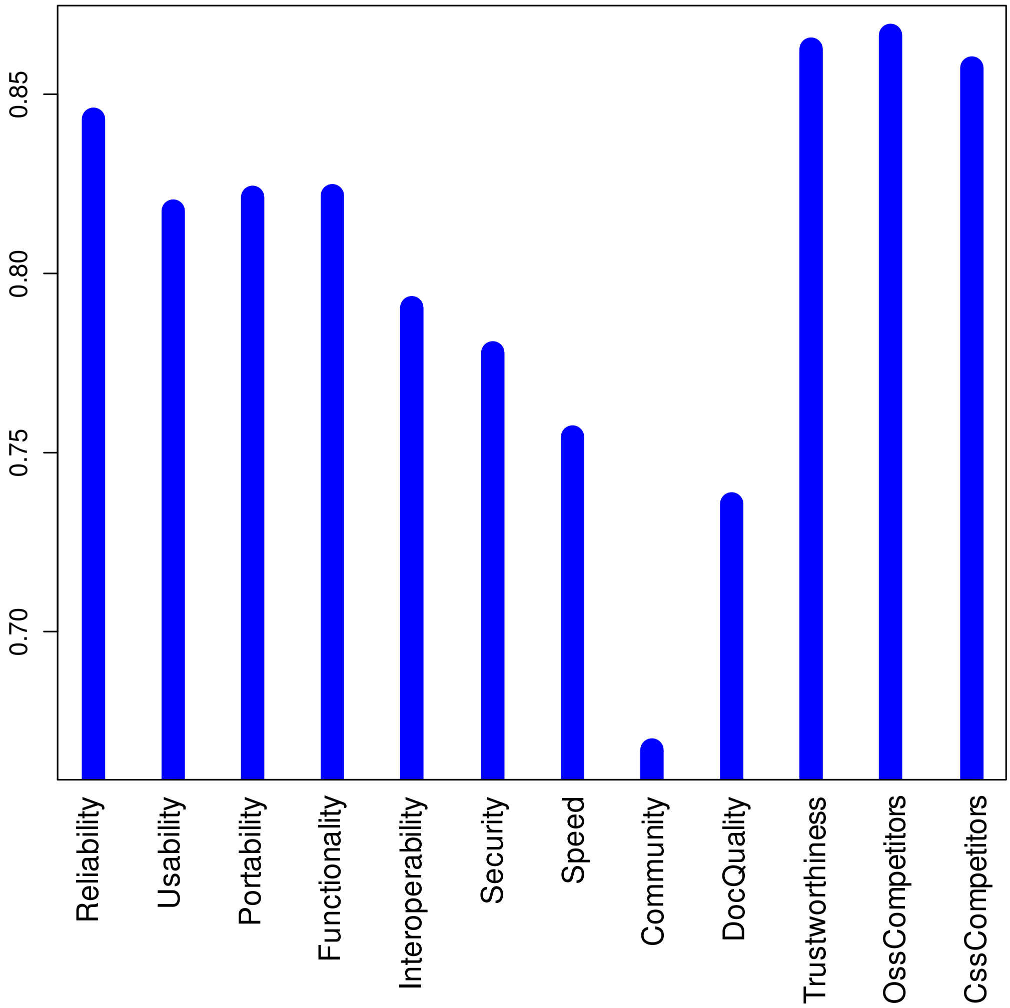

Figure 9. Median ratings of the evaluated qualities

It is possible to see that most users are quite satisfied with most qualities (including the overall trustworthiness of OSS). A noticeable exception is the level of support provided by the developer communities.

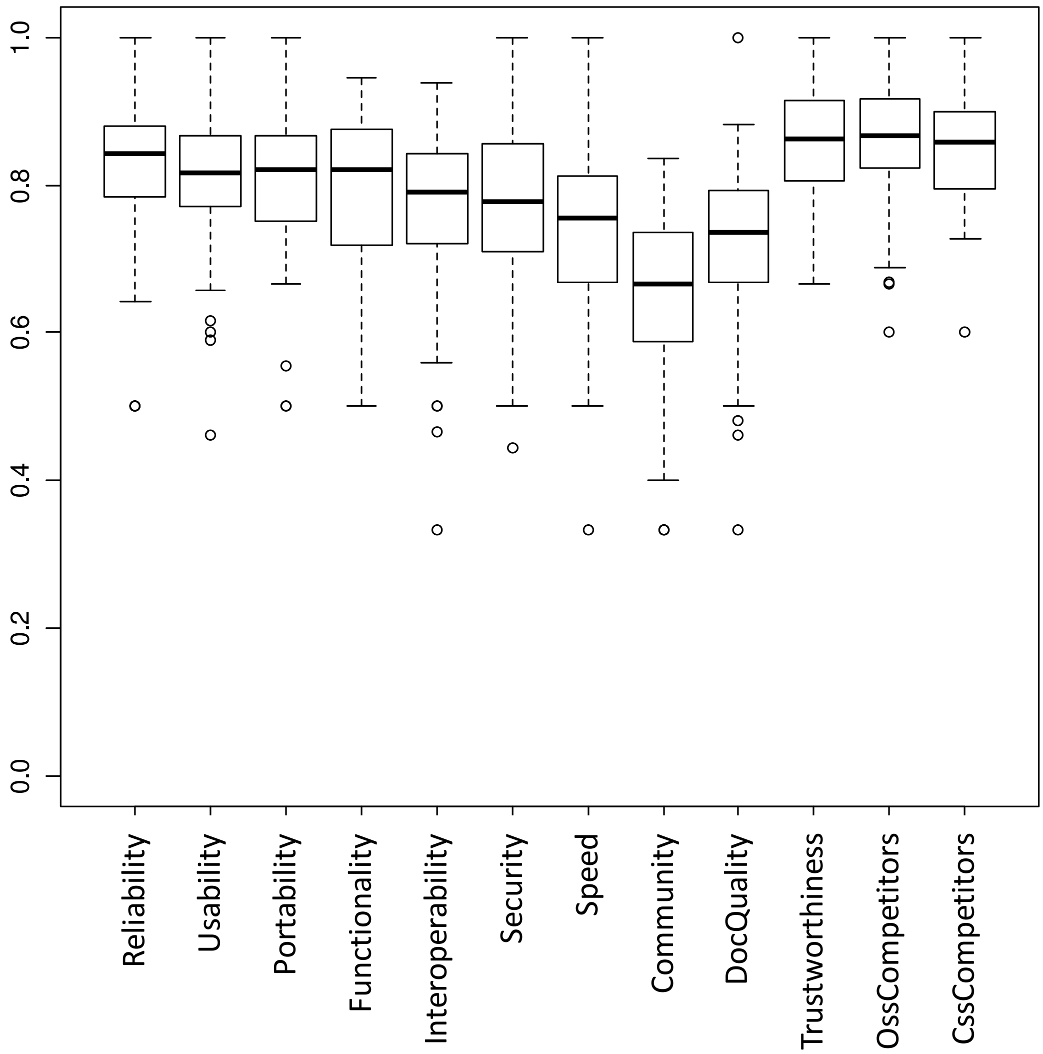

Figure 10. Distribution of the evaluated qualities across products

The overall quality of the evaluated OSS products is reported in Figure 11, where two types of overall quality are reported:

- One is the overall trustworthiness as reported by the users.
- The other is obtained as the sum of all positive grades (in all the considered sub-qualities) divided by the total number of grades.

It is possible to see that –with some exceptions– the two types of evaluations match reasonably well.

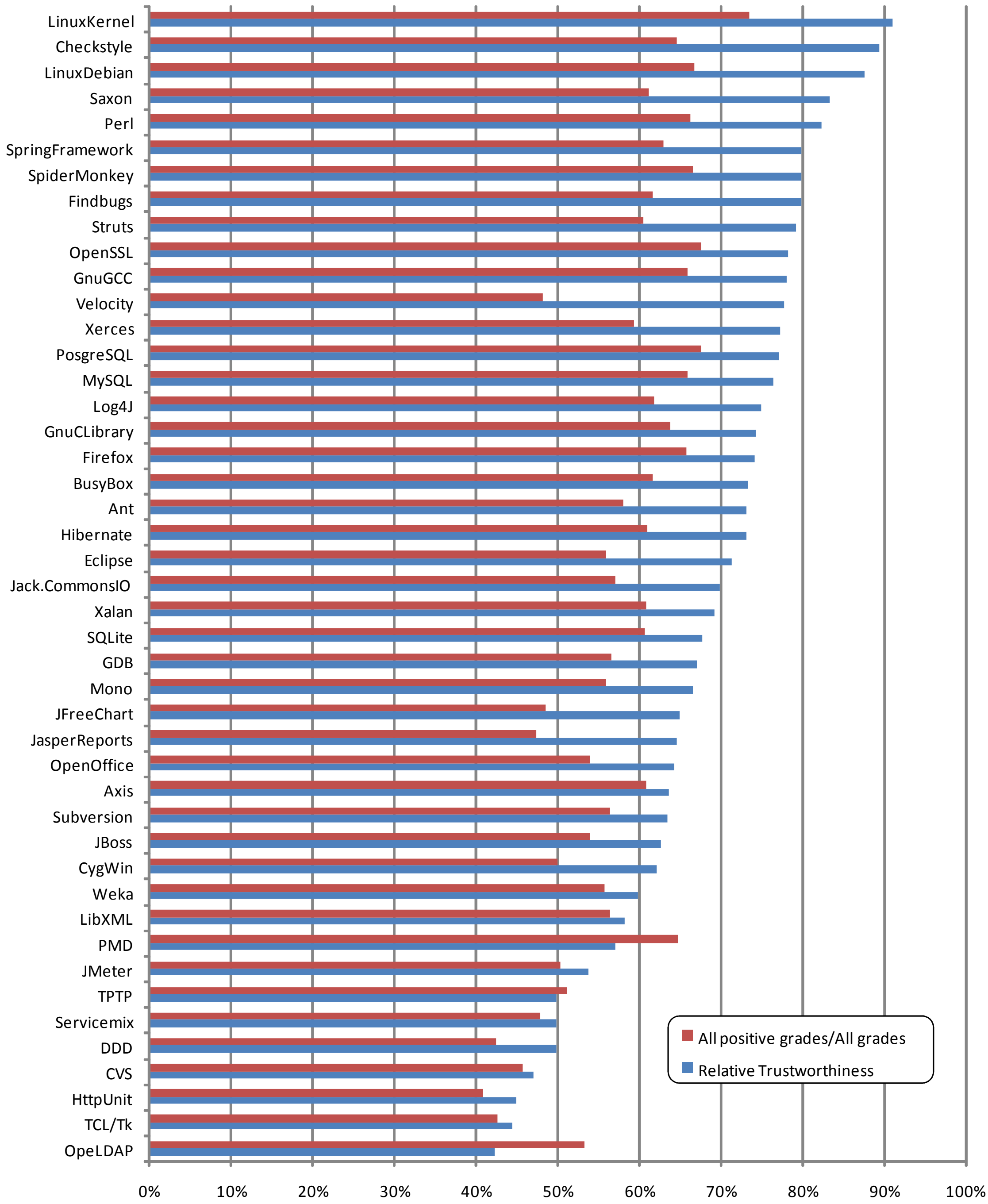

Figure 11. Quality of products

Another bit of analysis that was carried out outside the scope of Task 5.6.2 aims at understanding which qualities affect most the perceived overall trustworthiness of OSS products.

Analysis based on Ordinary Least Squares regression yielded a few models, which in general confirm that the investigated qualities do affect trustworthiness.

Below a couple of these models are synthetically reported[2]. The first one indicates that trustworthiness is proportional to the level of satisfaction of

[2] A detailed guide to the interpretation of the models' parameters is reported in [18].

functional requirements and to the security level of the product. The second indicates that trustworthiness is proportional to the level of satisfaction of functional requirements and to the efficiency of the product.

Both these results are quite expected, and confirm that the collected data are able to reflect the users' feelings.

```
=====================================================================
Trustworthiness  vs.  Functionality, Security
Residuals p-value 0.1152708
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 0.01988826 0.07911946 0.2513700 0.8037603311
x1          0.62797435 0.13954483 4.5001622 0.0001618049
x2          0.47198426 0.18052049 2.6145744 0.0154954569
Adj. R2 =  0.7032103
Eliminati: 9 / 35
MMRE =  19.38458
Pred(25) =  77.14286
Error range = [ -96.02235 .. 117.5477 ]
=====================================================================
=====================================================================
Trustworthiness  vs.  Functionality, Speed
Residuals p-value 0.2738275
              Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 0.08605156 0.05424757 1.586275 1.257661e-01
x1          0.61597188 0.07811995 7.884950 4.074454e-08
x2          0.47726646 0.09735174 4.902496 5.323043e-05
Adj. R2 =  0.7985336
Eliminati: 8 / 35MMRE =  15.28467
Pred(25) =  80
Error range = [ -82.78969 .. 80.18574 ]
=====================================================================
```

5 OBJECTIVE EVALUATIONS OF OSS PRODUCT CHARACTERISTICS

In this section we report about the usage of the QualiPSo tools that evaluate different characteristics –both static and dynamic– of OSS products.

5.1 Static code measurement

5.1.1 Static measure of Java code

The static characteristics of Java code were measure using the MacXim QualiPSo tool.

A synthesis of the measures concerning the size and structure of programs is reported in Table 3 (more specific measures, such as the number of private or protected methods, have been omitted for simplicity).

Table 3. Size and structure measures

	eLOC	Num. comment lines	Num. packages	Num. classes	Num. Abst. Classes	Num. interf.	Num. methods	Num. attributes
Min	229	110	1	4	5	1	25	22
Max	203545	187944	505	4678	199	514	42833	27528
Mean	59125	54021	71	1141	57	155	12199	5844
Median	41216	38061	39	994	45	131	11608	5121
Stdev	58262	52527	112	1073	44	141	11103	6459

Typical object-oriented measures (namely those proposed by Chidamber and Kemerer [14]) and complexity measures (McCabe [15]) are reported in Table 4.

Table 4. McCabe and Chidamber&Kemerer measures

	McCabe	CBO	LCOM	DIT	NOC	RFC
Min	1.2	0.7	3.3	1.0	0.0	8.0
Max	4.0	55.0	1038.1	1.5	1.7	31.0
Mean	2.1	6.2	382.6	1.2	0.9	19.1
Median	2.1	4.0	307.9	1.1	0.9	18.3
Stdev	0.6	11.3	334.4	0.1	0.5	4.9

5.1.2 Static measure of C++ code

The static characteristics of Java code were measure using the Kalibro QualiPSo tool.

A synthesis of the measures concerning the size and structure of programs is reported in Table 5 (more specific measures, such as the number of abstract classes, have been omitted for simplicity).

Table 5. Size and structure measures

	eLOC	Num. modules	Num. methods	Num. Attributes
Min	14532	108	918	858
Max	8106513	13601	319352	433922
Mean	970398.5	2721.2	33194.9	40522.9
Median	378580	1536	10945	8178
Stdev	2013580.0	3709.9	79716.2	109867.7

Typical object-oriented measures (namely those proposed by Chidamber and Kemerer [14]) are reported in Table 7.

Table 6. Chidamber&Kemerer measures

	CBO	DIT	NOC	LCOM	RFC
Min	0.0	0.0	0.0	2.1	5.6
Max	12.5	0.6	0.3	10.8	188.4
Mean	4.9	0.1	0.0	5.4	66.2
Median	4.6	0.0	0.0	3.9	42.6
Stdev	3.3	0.2	0.1	2.9	58.9

5.1.3 Evaluation of code well formedness and style

ECA (Elementary Code Assessment) rules prescribe conditions that should be ideally be always satisfied by source code. In fact, the violation of these rules indicates the probability of errors; i.e., code characterized by several violations is expected to be quite error-prone. Of course, it is hardly possible to state that whenever a violation occurs a malfunction will take place; nevertheless, the analysis carried out in Task 5.6.2 demonstrated that there is a correlation between the perceived reliability and the number of ECA rule violations.

In QualiPSo the ability of evaluating ECA rules provided by tools like PMD and Checkstyle was incorporated in MacXim. Specifically, the following ECA rules are currently supported by QualiPSo tools (the terminology is borrowed from PMD):

1. Avoid Catching Throwable
2. Constructor Calls Overridable Method
3. Class Naming Conventions
4. Empty Catch Block
5. Excessive Class Length
6. Excessive Method Length
7. For Loops Must Use Braces
8. If Else Statements Must Use Braces
9. If Statements Must Use Braces
10. Missing Break In Switch
11. Override Both Equals And Hashcode
12. Unused Private Field
13. Unused Private Method
14. Switch Statements Should Have Default
15. Use Equals To Compare Strings

16. While Loops Must Use Braces

The set of considered rules addresses both

- situations that are very likely to cause run-time troubles (rules 1, 2, 4, 11 and 15)
- simple stylistic issues, which are less likely to result in malfunctions.

Accordingly, we studied the OSS products with respect to the number of critical rule violations and the total number of rule violations. It was possible to see that –with the exception of Struts– all products feature a reasonably low level of rule violations per effective lines of code. Density of total ECA rule violations in the examined OSS products (computed as the total number of ECA rule violations divided by the number of effective LOC) is illustrated in Figure 12.

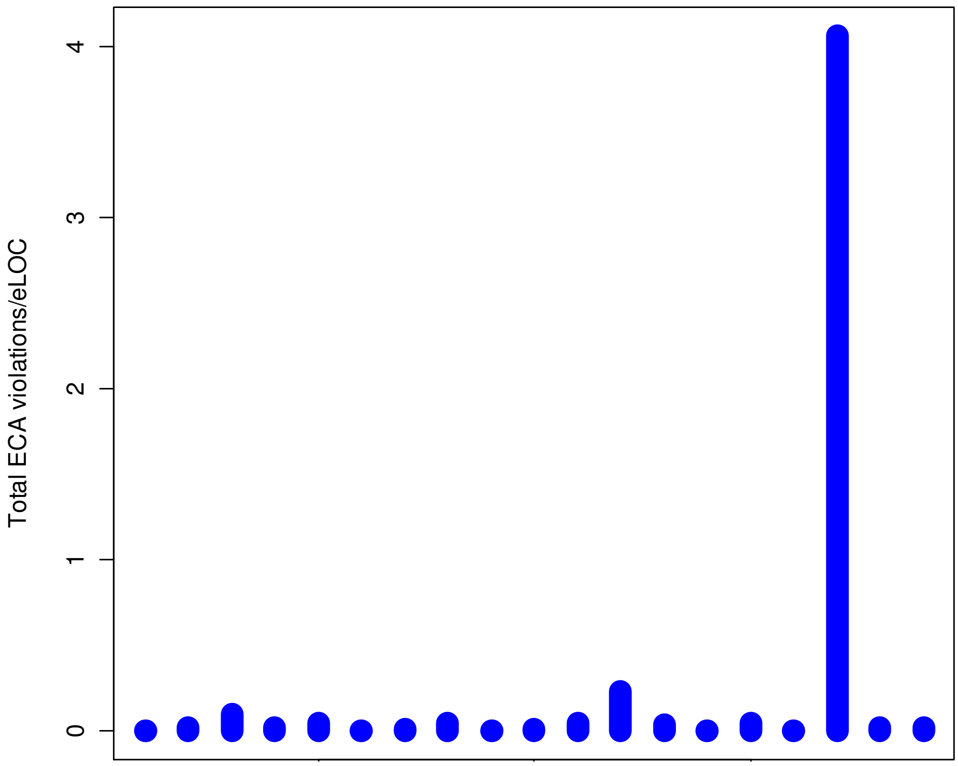

Figure 12. Density of total ECA rule violations in the examined OSS products

The situation changes when only critical rule violations are concerned: there are half a dozen products that feature a density of violations greater than advisable (Figure 13).

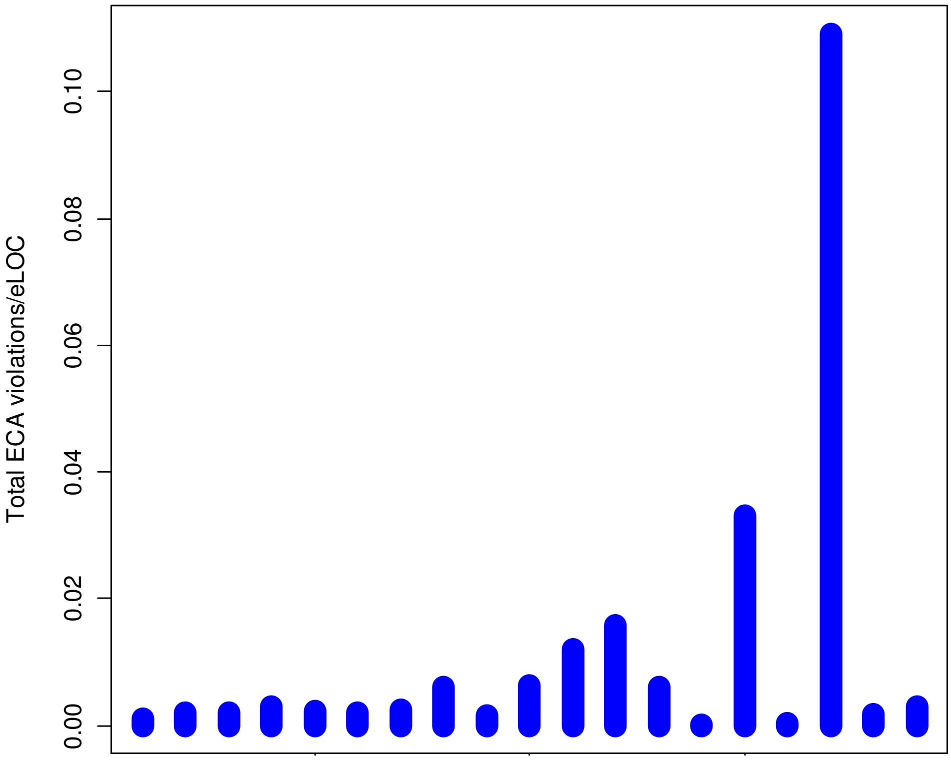

Figure 13. Density of critical ECA rule violations in the examined OSS products

It must be noted that considering the density of violations is necessary to get an indications of how “good” is a product from the point of view of developers.

The absolute number of violations (see Figure 14) conveys more interesting information from the point of view of the user perception of quality: in fact, the more rule violations, the more probable are user-perceivable failures. Notice that Struts (the third product from the right) does not appear likely to cause many failures (even though they are caused by faults located in relatively small code).

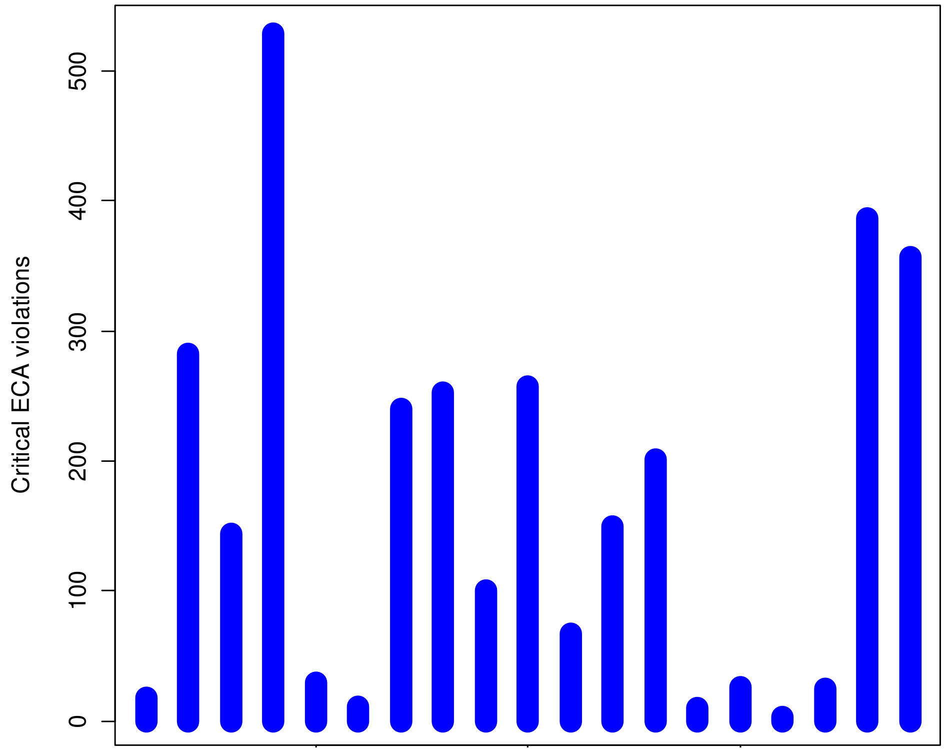

Figure 14. Number of critical ECA rule violations in the examined OSS products

5.2 Analysis of product development

Measures about product development were collected by means of the StatSVN QualiPSo tool.

A synthesis of the measures concerning product development is reported in Table 7, Table 8 and Table 9.

Table 7. Product level measures of product development

	Num. developers	Major releases per year	Minor releases per year
Min	1	0	0
Max	43	2	11
Mean	18	0	3
Median	17	0	2
Stdev	12	1	3

Table 8. File level measures of product development

	Num. files	Files added per year	Files removed per year	Revisions per file
Min	2	1	0	2
Max	8183	4925	3239	13
Mean	2616	773	439	7
Median	2256	320	139	6
Stdev	2227	1164	774	3

Table 9. LOC level measures of product development

	Commits per year	LOC added per year	LOC deleted per year	LOC changed per year
Min	3	0	0	0
Max	8895	272754	162009	110744
Mean	2529	44954	30324	19429
Median	1852	8350	2379	5971
Stdev	2448	77760	53291	30196

5.3 Analysis of licensing information

Measures about licensing information reported in OSS products were collected by means of the OSLC QualiPSo tool.

A synthesis of the collected measures is reported in Table 10.

Table 10. Measures of product licensing information

	Copyrighted Files	Copyright Holders	Distinct Licenses	Global Conflicts	Reference Conflicts	Licensed Files	Uncertain Licensed Files	Unlicensed Files
Min	0	0	0	0	0	0	0	0
Max	2069	58	6	5	38	1636	2075	1650
Mean	428	9	2	1	4	503	189	136
Median	17	1	2	0	0	295	0	1
Stdev	654	18	1	1	11	518	515	390

5.4 Evaluation by testing tools

The level of coverage of the tests that are available for a given set of OSS products was measured by means of the Jabuti tool. Four structural testing criteria—namely, *all-Nodes*, *all-Edges*, *all-Uses*, and *all-Potential-Uses*— have been used to assess the thoroughness of functional requirements testing in OSS projects. To conduct the coverage analysis of the OSS projects we used JaBUTi – Java Byte-code Understanding Tool – a tool that statically analyzes bytecode compiled programs and obtains testing requirements with respect to the aforementioned testing criteria.

All-Nodes: refers to the execution of all statements of a product implementation at least once;

All-Edges: refers to a test set that makes each conditional statement assume true and false values at least once;

All-Uses: refers to a test set *T* to include tests that exercise paths without redefinitions of a variable *X* from every definition of *X* (a value assignment to *X*) to every subsequent use of *X* (a reference to *X*) (such paths are called def-clear paths with respect to *X*);

All-Potential-Uses: is a variation of *all-Uses* in which the test set *T* should include tests that exercise def-clear paths from every definition of *X* to any point of the program reachable by a def-clear path with respect to *X*. The idea is to check potential uses of *X*.

A synthetic view of the collected measures, computed on a set of 8 OSS projects, is reported in

Table 11 (*ei* and *ed* means exception-independent and the exception-dependent testing criteria).

Table 11. Test coverage measures

	All nodes *ei*	All nodes *ed*	All edges *ei*	All edges *ed*	All uses *ei*	All uses *ed*	All potential uses *ei*	All potential uses *ed*
min	19.73	0.33	16.58	0.09	15.62	0.23	14.79	0.19
Max	80.71	22.47	78.53	6.05	76.70	21.12	74.43	18.10
mean	44.65	9.93	39.43	2.65	38.13	9.87	36.09	8.10
median	37.90	7.47	31.73	2.05	32.20	8.55	30.76	6.62

6 MEASUREMENT DATA STORAGE

The data being collected by means of measurements, interviews, from other data sources, etc., are stored in a well-structured, persistent repository that supports the analysis activities performed in the context of Task 5.6.2.

The repository also integrates nicely with the measurement and data collection tools.

The repository collects data from various Qua;iPSo tools and makes them available to the analysis activities and to the reporting tool (Spago4Q), as shown in Figure 15.

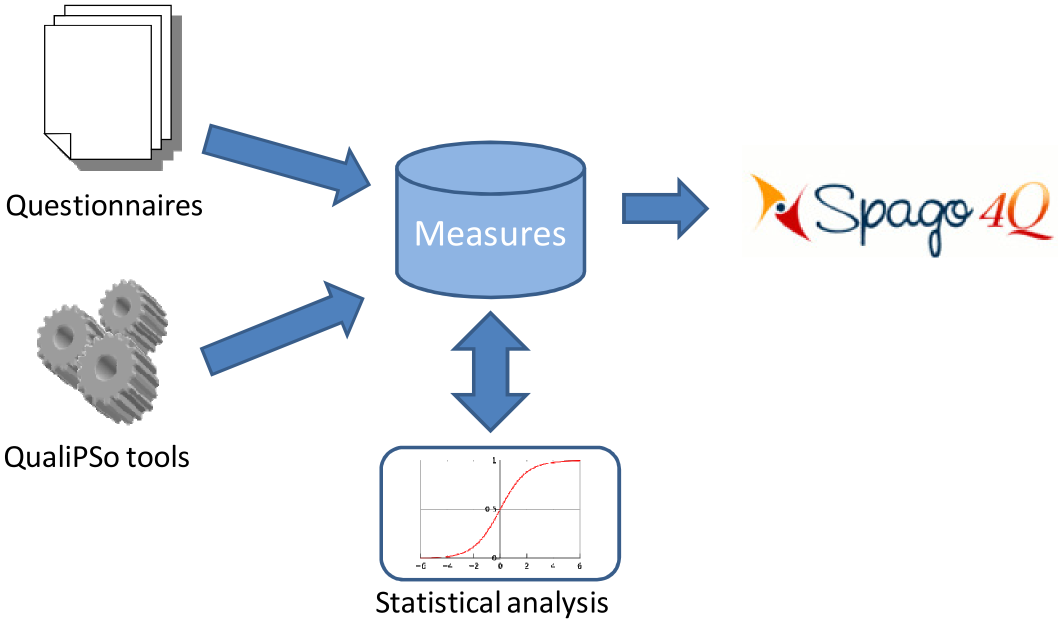

Figure 15. Role of the measures repository.

The repository is based on the MySQL relational DBMS. MySQL was chosen because it is a reliable OS product and because it had already been used in conjunction with Spago4Q.

The database design activity is illustrated in Figure 16 and Figure 17.

In particular, Figure 16 accounts for the Tables that are dedicated to storing the user perception of the trustworthiness of the OSS products. Table OSS_Product stores the data concerning the OSS products (name, version, licence, etc.); table User stores a set of data that characterize the users that provided the trustworthiness evaluations; table PerceivedTrustworthiness has an attribute for every quality aspect (reliability, safety, usability, etc.) that is relevant to characterize the trustworthiness of OSS products.

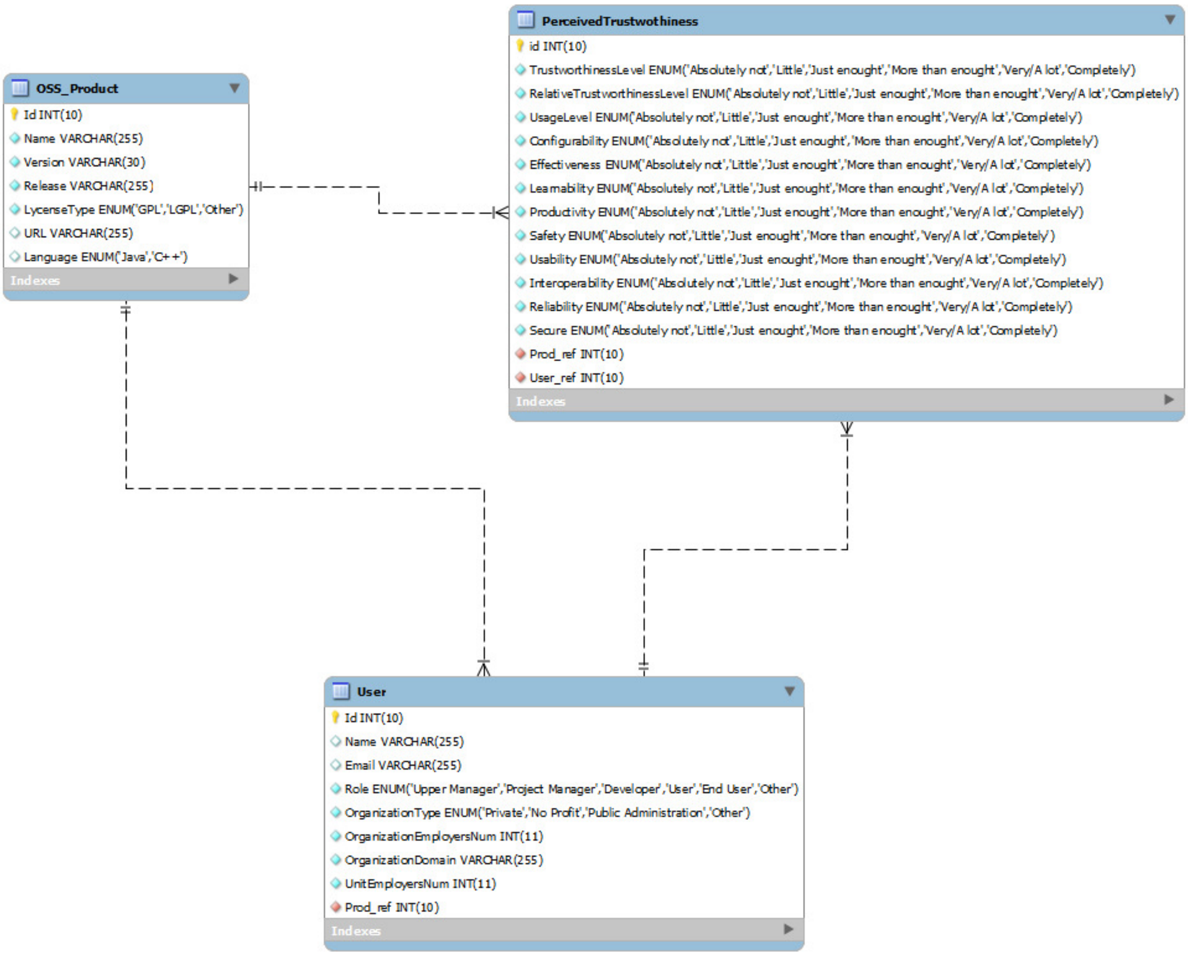

Figure 16. Conceptual model including all user perceived aspects of trustworthiness.

Figure 17 illustrates the tables that were designed to contain the data concerning the objective measures of the product characteristics. There is a table for each element (class, method, attribute, ...) and granularity level (application, package, class, ...) for which measures can be defined. Besides such tables, there are three tables for storing the measures form the non-QualiPSo tools (PMD, FindBugs, Checkstyle, ...) that could be used used. Finally, there is a table for storing data from any additional measurement tool that one could decide to use in the future.

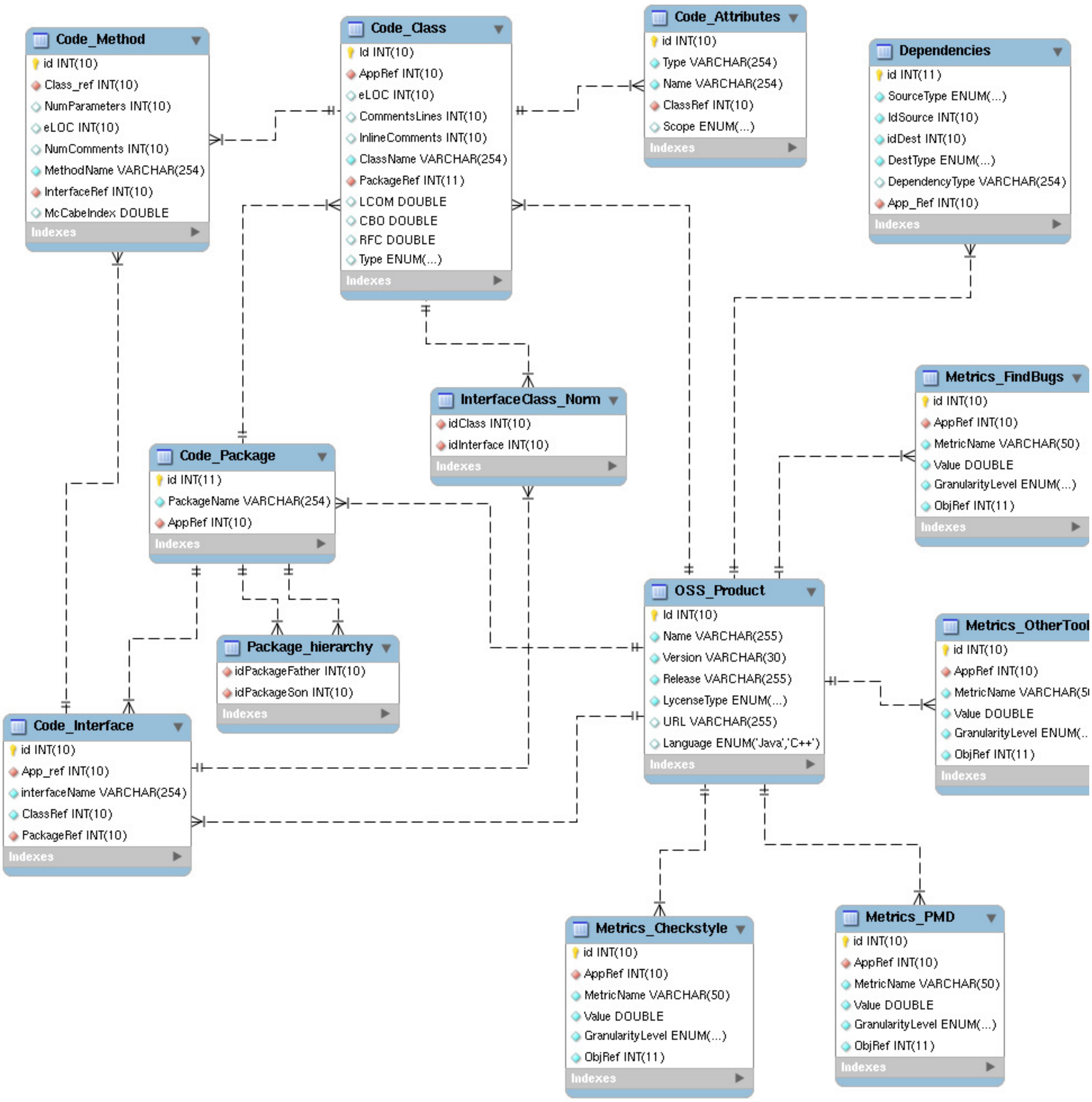

Figure 17. Conceptual model including the objective data.

7 ANALYSIS OF COLLECTED DATA

The approach to data analysis and the achieved results are described in [18]. Here we report only a few indications about the tools used to carry out the analysis.

The analysis of the collected data is an activity that is carried out off-line with respect to the QualiPSo platform, and aims at deriving quantitative models of OSS trustworthiness. The parameters of the valid models identified are then embedded into the QualiPSo platform, so that trustworthiness evaluation can be performed upon request according to the models.

In order to perform statistical analysis it was not necessary to build an ad-hoc tool, since there were already several OS tools supporting statistical computations. In QualiPSo we just had to customize one of such tools in order to make it suitable for the type of analysis we had in mind (i.e., logistic regression: see [18] for a bit of discussion about it).

R [18] was chosen because it is a powerful, mature tool, licensed under the GPL license. Moreover, R is programmable: this made it relatively easy to build the analysis programs needed. It was also possible to exploit the numerous libraries provided by R, while we could implement the feature (e.g., the computation of R_{log}) not natively supported.

The experimentation proceeded as follows:

1) The data to be analyzed were exported from the data repository into a format that could be easily inspected and –if needed– modified by the analyzer. We chose the universally supported comma-separated values (CSV) format.
2) The R code for reading and analyzing the code was written.
3) The code was tested in interactive mode. In this mode R works as an interpreter of the code. It is possible to stop the computation at any point and inspect partial results.
4) The code was finally run in batch mode. The results were saved into text files and –as far as graphs were concerned– jpeg files.

Figure 18 shows R at work with our analysis code. The window on the left hand side shows the code being executed (and the textual outputs, if any); the window on the right hand side reports the graphic output.

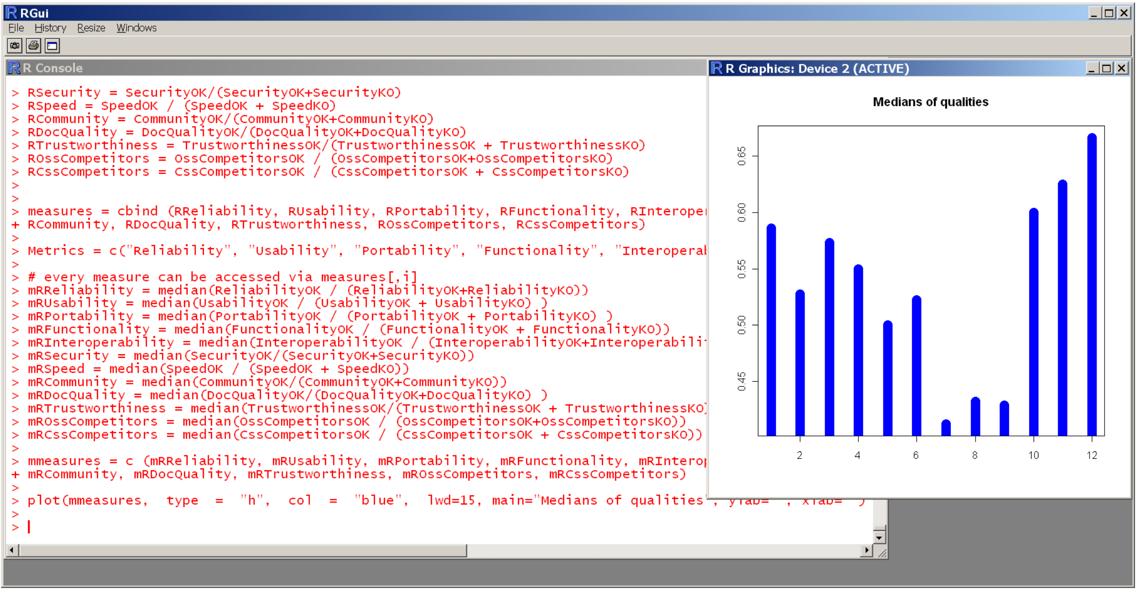

Figure 18: A data analysis session using R.

The analysis carried out hade two main purposes:

- Verifying the correctness of data and gaining a better understanding of the data themselves.
- Building the quantitative trustworthiness models.

The results of the latter activity are reported in detail in [18].

8 CONCLUSIONS

In this working document, we reported the activities performed in the second round of experiments in the context of Task 5.6.1 - Experimentation on the trustworthiness of Open Source Software. The results of such activities provided the data concerning the qualities and features of OSS products that were analyzed in Task 5.6.2 - Model building [18].

9 REFERENCES

[1] V. Basili and D. Weiss, "A methodology for collecting valid software engineering data," IEEE Transactions on Software Engineering, vol. SE-10, no. 6, pp. 728-738, 1984.

[2] V. Basili and D. Rombach, "The TAME project: towards improvement-oriented software environments," IEEE Transactions on Software Engineering, vol. 14, no. 6, pp. 758-773, 1988.

[3] V. Basili, G. Caldiera, and D. Rombach, "Goal/Question/Metric Paradigm," in En-cyclopedia of Software Engineering, vol. 1, J. C. Marciniak, Ed.: John Wiley & Sons, 1994, pp. 528-532.

[4] R. van Solingen and E. Berghout, The Goal/Question/Metric Method, McGraw-Hill, 1999. http://www.gqm.nl/

[5] V.R. Basili, G. Caldiera, H.D. Rombach, "The Goal Question Metric Approach", http://www.cs.umd.edu/projects/SoftEng/ESEG/papers/gqm.pdf

[6] V. del Bianco, M. Chinosi, L. Lavazza, S. Morasca, D. Taibi, "How European software industry perceives OSS trustworthiness and what are the specific criteria to establish trust in OSS", QualiPSo report, October 2007, http://qualipso.semanticdesktop.org/xwiki/bin/view/Wiki/WP51.

[7] V. del Bianco, L. Lavazza, S. Morasca, D. Taibi, "Analysis of relevant open-source projects and artifacts", Working document wd 5.2.1, Version 0.1 - QualiPSo report, February 2008.

[8] V. del Bianco, L. Lavazza, S. Morasca, D. Taibi, "Definition of trustworthiness of software products and artefacts", Working document wd 5.3.1, Version 2.0 - QualiPSo report, October 2008.

[9] V. del Bianco, L. Lavazza, S. Morasca, D. Taibi, "Identification of factors that influence the trustworthiness of software products and artefacts", Working document wd 5.3.2, Version 2.0 - QualiPSo report, October 2008.

[10] L. Lavazza, S. Morasca, D. Taibi, D. Tosi, "Definition of trustworthiness of software products and artefacts", Working document wd 5.3.1, Version 3.0 - QualiPSo report, October 2009.

[11] L. Lavazza, S. Morasca, D. Taibi, D. Tosi, "Definition of trustworthiness of software products and artefacts", Working document wd 5.3.1, Version 4.0 - QualiPSo report, January 2011.

[12] ISO/IEC 9126-1:2001 "Software Engineering—Product Quality—Part 1: Quality model", June 2001.

[13] L. Lavazza, "Providing automated support for the GQM measurement process", IEEE Software, vol. 17, n. 3, May-June 2000.

[14] Chidamber, S. R. and Kemerer, C. F. 1994. A Metrics Suite for Object Oriented Design. *IEEE Trans. Softw. Eng.* 20, 6 (Jun. 1994), 476-493.

[15] T.J. McCabe, "A complexity measure", IEEE Transactions on Software Engineering, vol.2, n.4, December 1976.

[16] V. del Bianco, L. Lavazza, S. Morasca, D. Taibi, "Experimentation on the trustworthiness of Open Source Software, v. 1", Working document wd 5.6.1, Version 1.0 - QualiPSo report, October 2008.

[17] L. Lavazza, S. Morasca, D. Taibi, D. Tosi, "Experimentation on the trustworthiness of Open Source Software, v. 2", Working document wd 5.6.1, Version 2.0 - QualiPSo report, June 2009.

[18] L. Lavazza, S. Morasca, D. Taibi, D. Tosi, "Trustworthiness models for Open Source Software", Working document wd 5.6.2, Version 3.0 - QualiPSo report, January 2011.

[19] http://www.r-project.org/

10 APPENDIX THE QUESTIONNAIRE FOR ASSESSING THE PERCEIVED TRUSTWORTHINESS OF OSS

Here follows the questionnaire for evaluating the users' perceived trustworthiness.

YOUR OPINION WILL BE VERY USEFUL TO THE OSS COMMUNITY

Qualipso Survey – The Trustworthiness of Open Source Product

www.qualipso.org

Why This Survey?

The purpose of this survey is to elicit information from the users and developers of Open Source Software (OSS) products about their perceptions on the trustworthiness of OSS products and the related factors.

Who Are We?

This survey has been developed in the framework of the QualiPSo (Quality Platform for Open Source Software) project, which is a European Union-funded Integrated Project which aims at making a major contribution to the state of the art and practice of Open Source Software. The QualiPSo project started in November 2006 and will last until October 2010. The project brings together 18 software companies, application solution developers, and research institutions. Its goal is to define and implement technologies, procedures, and policies to leverage the Open Source Software development current practices to sound, well-recognized, and established industrial operations.

What Will Happen to the Questionnaires?

All information provided by each individual or organization will be treated as confidential. As such, it will not be released in other form than aggregated statistical analyses that will make it impossible to identify the single respondents.

Please do not hesitate to contact us if you need any information or clarification.

Sandro Morasca
Università degli Studi dell'Insubria
Dipartimento di Scienze della Cultura,
Politiche e dell'Informazione
Via Carloni 78
I-22100 Como, Italy
sandro.morasca@uninsubria.it

Some questions may not be applicable to you: just skip them. When you answer, *please always give <u>your</u> personal opinion*.

Your name (optional) ______________________________

Your email address (optional) ______________________________

Your role in your organization ☐ Upper Manager ☐ Project manager ☐ Developer ☐ Other

Type of organization ☐ private ☐ no profit ☐ Public Administration

Number of employees of your organization: ______________________________

Organization's domain(s) (Public Administration, banking/finance, …)

____

Number of employees of your specific unit in your organization: ______________________________

Unit's domain(s) (Public Administration, avionics, banking/finance, …, same as organization's):

____

Your use of the OSS product **Java Projects**	Checkstyle	Eclipse	Findbugs	Hibernate	HttpUnit	Jack.Commons IO	Jasper Reports	JBoss	JFreeChart	JMeter	Log4J	PMD	Saxon	Spring Framework	Struts	Tapestry	TPTP	Velocity	Weka	Xalan	Xerces	Servicemix
Do you use the product?																						
Yes																						
Maybe in the future																						
No																						
What version of the product are you using																						
The last one																						
A recent one																						
What is your relationship with the OSS product																						
User of the product 'as is'																						
Integrator/customizer																						
Producer																						
Other																						

Please give us **your opinion** for the projects you are familiar with, by ranking the factors below on a 1 to 6 scale, where 1= absolutely not; 2=little; 3=just enough; 4=more than enough; 5= very/a lot; 6= completely

Just skip the projects you are not familiar with

Quality of the OSS product **Java Projects**	Checkstyle	Eclipse	Findbugs	Hibernate	HttpUnit	Jack.Commons IO	Jasper Reports	JBoss	JFreeChart	JMeter	Log4J	PMD	Saxon	Spring Framework	Struts	Tapestry	TPTP	Velocity	Weka	Xalan	Xerces	Servicemix
How **familiar** are you with the product?																						
How **usable** is the product?																						
How **portable** is the product?																						
How much does/did the product **satisfy** your **functional requirements** when you use/used it?																						
How **interoperable** is the product?																						
How **reliable** is the product?																						
How **secure** is the product?																						
How useful is the **product developer community** to you?																						
How **fast** is the product?																						
How **well documented** is the product?																						

Based on your answers to the questions above:

How much do you **trust** the product, overall?																						
How much do you **trust** the product, **compared to its Open Source competitors**?																						
How much do you **trust** the product, **compared to its non Open Source competitors**?																						

Qualipso Quality Platform for Open Source Software

OW2 Consortium

Information Society Technologies

Please tick the correct answer.

Your use of the OSS product **C/C++ Projects**	Ant	Axis	BusyBox	CVS	CygWin	DDD	GDB	Gnu C Library	Gnu GCC	Lib XML	Linux Kernel	Mono	MySQL	OpeLDAP	Open Pegasus	Open SSL	Perl	PosgreSQL	SpiderMonkey	SQLite	Subversion	TCL/Tk
Do you use the product?																						
Yes																						
Maybe in the future																						
No																						
What version of the product are you using																						
The last one																						
A recent one																						
What is your relationship with the OSS product																						
User of the product 'as is'																						
Integrator/customizer																						
Producer																						
Other																						

Please give us **your opinion** for the projects you are familiar with, by ranking the factors below on a 1 to 6 scale, where 1= absolutely not; 2=little; 3=just enough; 4=more than enough; 5= very/a lot; 6= completely

Just skip the projects you are not familiar with

Quality of the OSS product **C/C++ Projects**	Ant	Axis	BusyBox	CVS	CygWin	DDD	GDB	Gnu C Library	Gnu GCC	Lib XML	Linux Kernel	Mono	MySQL	OpeLDAP	Open Pegasus	Open SSL	Perl	PosgreSQL	SpiderMonkey	SQLite	Subversion	TCL/Tk
How **familiar** are you with the product?																						
How **usable** is the product?																						
How **portable** is the product?																						
How much does/did the product **satisfy** your **functional requirements** when you use/used it?																						
How **interoperable** is the product?																						
How **reliable** is the product?																						
How **secure** is the product?																						
How useful is the **product developer community** to you?																						
How **fast** is the product?																						
How **well documented** is the product?																						

Based on your answers to the questions above:

How much do you **trust** the product, overall?																						
How much do you **trust** the product, **compared to its Open Source competitors**?																						
How much do you **trust** the product, **compared to its non Open Source competitors**?																						